

PLP - Primer Parcial - 2^{do} cuatrimestre de 2019

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido, número de orden y cantidad de hojas en la primera hoja, y numerar las hojas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.

Ejercicio 1 - Programación funcional

En este ejercicio trabajaremos con listas infinitas representadas con funciones de la siguiente manera: representaremos a la lista $[x_0, x_1, x_2, \dots]$ con una función l tal que $l(i) = x_i$, es decir, con una función que toma una posición y devuelve el elemento que ocupa esa posición en la lista (observemos que comenzamos contando desde 0). A menos que se especifique lo contrario, **no está permitido**: el uso de recursión explícita, ni el operador (!!) ni la conversión entre esta representación de listas y la usual (en cualquiera de los dos sentidos). Consecuentemente, tampoco se pueden definir funciones que imiten o incluyan estos comportamientos. Ante la duda, consultar con los docentes. Pueden usar los ejercicios de la práctica o los vistos en clase, colocando referencias claras. Dar el **tipo** de todas las funciones definidas.

Definimos el siguiente renombre de tipos: $\text{type LI } a = \text{Int} \rightarrow a$

a. Definir las siguientes expresiones:

- i) **naturales** :: LI Int (la lista de todos los naturales en orden, incluyendo el 0)
- ii) **pares** :: LI Int (la lista de todos los números pares en orden)
- iii) **repeatI** :: [a] \rightarrow LI a, que resulta en una lista infinita a partir de repetir, en orden, una lista finita. Para esta definición puede utilizarse el operador (!!).
Ejemplo: $\text{repeatI } [1, 2, 3] \equiv [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, \dots]$.

b. i) Definir los siguientes operadores sobre listas infinitas, análogos a los que conocemos para las listas usuales:

- 1) **consI** :: a \rightarrow LI a \rightarrow LI a (el constructor de listas que agrega un elemento al principio de una lista)
- 2) **headI** :: LI a \rightarrow a
- 3) **tailI** :: LI a \rightarrow LI a

ii) Definir los siguientes esquemas de recursión sobre listas infinitas, análogos a los que conocemos para las listas usuales:

- 1) **foldI** :: (a \rightarrow b \rightarrow b) \rightarrow LI a \rightarrow b (no proveemos un caso base ya que, como la lista es infinita, la cola nunca es vacía. Para esta definición puede utilizarse recursión explícita)
- 2) **mapI** :: (a \rightarrow b) \rightarrow LI a \rightarrow LI b
- 3) **filterI** :: (a \rightarrow Bool) \rightarrow LI a \rightarrow LI a (puede suponerse que el resultado es infinito)

c. Definir la expresión **multiplos** :: LI (LI Int), que representa la lista (en orden) de las listas de todos los múltiplos de cada número natural. Es decir,

$$\text{multiplos} \equiv [[0, 0, 0, \dots], [0, 1, 2, 3, \dots], [0, 2, 4, 6, \dots], [0, 3, 6, 9, \dots], \dots]$$

(notar que es una lista infinita de listas infinitas).

Ejercicio 2 - Cálculo Lambda Tipado

En este ejercicio consideramos la extensión para listas trabajada en la práctica.

$$\sigma ::= \dots \mid [\sigma] \quad M ::= \dots \mid []_\sigma \mid M :: M$$

$$\frac{}{\Gamma \triangleright []_\sigma : [\sigma]} \text{(T-EMPTY)}$$

$$\frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : [\sigma]}{\Gamma \triangleright M :: N : [\sigma]} \text{(T-APPEND)}$$

$$\frac{M \rightarrow M'}{M :: N \rightarrow M' :: N} \text{(E-APPEND-1)}$$

$$\frac{N \rightarrow N'}{V :: N \rightarrow V :: N'} \text{(E-APPEND-2)}$$

Además, agregaremos términos para representar listas por comprensión, con un selector y una guarda, de la siguiente manera: $[M \mid x \leftarrow S, P]$, donde x es el nombre de una variable que puede aparecer libre en los términos M y P . La semántica es análoga a la de Haskell: para cada valor de la lista representada por el término S , se sustituye x en P y, de resultar verdadero, se agrega M con x sustituido al resultado.

Por ejemplo: $[succ(n) \mid n \leftarrow 0 :: 1 :: 0 :: 2 :: \llbracket_{Nat}, not\ isZero(n) \rrbracket] \rightsquigarrow succ(1) :: succ(2) :: \llbracket_{Nat}$ (suponiendo not definido como una macro de una función que computa la negación).

a. i) Dar las reglas de tipado para soportar los nuevos términos.

ii) Dada la regla de tipado para *even* :
$$\frac{\Gamma \triangleright M : Nat}{\Gamma \triangleright even(M) : Bool} \text{ (T-EVEN)}$$

demostrar el siguiente juicio de tipado:

$$\emptyset \triangleright [\lambda x : Nat. n \mid n \leftarrow 0 :: 1 :: \llbracket_{Nat}, even(n) \rrbracket] : [Nat \rightarrow Nat]$$

b. i) Describir el nuevo conjunto de valores y dar las reglas de reducción en un paso para los nuevos términos.

ii) Reducir el término del ítem aii, omitiendo los pasos que solo involucran reducciones de *even* (*even* reduce a *True* si el número es par y a *False* si no). Puede abreviarse la escritura de algunos subtérminos que no intervengan en pasos posteriores de la reducción.

c. Definir como macros las siguientes funciones que se comportan como las homónimas de Haskell:

i) $filter_\sigma$ de tipo $(\sigma \rightarrow Bool) \rightarrow [\sigma] \rightarrow [\sigma]$

ii) $map_{\sigma\tau}$ de tipo $(\sigma \rightarrow \tau) \rightarrow [\sigma] \rightarrow [\tau]$

Ejercicio 3 - Inferencia de Tipos

Se desea modificar el algoritmo de inferencia para soportar el cálculo extendido con intervalos de números naturales. Se extenderán los tipos y términos de la siguiente manera:

$$\sigma ::= \dots \mid \text{Intervalo} \quad M ::= \dots \mid [M, M] \mid \text{case } M \text{ of } [] \rightsquigarrow M; x :: y \rightsquigarrow M$$

donde $[M, N]$ es el intervalo que va desde el número M hasta N y $\text{case } M \text{ of } [] \rightsquigarrow N; x :: y \rightsquigarrow O$ es un observador que recorre el intervalo M como si fuese una lista: con un caso para el intervalo vacío (N) y otro para un intervalo con al menos un elemento (O). En el segundo caso, las apariciones libres de la variable x en O se ligarán al primer elemento del intervalo y las apariciones libres de y en O se ligarán al intervalo sin su primer elemento.

La reglas de tipado son las siguientes:

$$\frac{\Gamma \triangleright M : Nat \quad \Gamma \triangleright N : Nat}{\Gamma \triangleright [M, N] : Intervalo} \quad \frac{\Gamma \triangleright M : Intervalo \quad \Gamma \triangleright N : \sigma \quad \Gamma, x : Nat, y : Intervalo \triangleright O : \sigma}{\Gamma \triangleright \text{case } M \text{ of } [] \rightsquigarrow N; x :: y \rightsquigarrow O : \sigma}$$

a. Dar la modificación del algoritmo necesaria para soportar la nueva extensión.

b. Aplicar el algoritmo extendido para tipar la siguiente expresión:

$$\text{case } (\lambda x. [0, x]) \text{ y of } [] \rightsquigarrow (\lambda x. [y, y]); x :: y \rightsquigarrow \lambda t. y$$

Ejercicios teóricos

a. Definir a la función *filter* sobre listas en términos del operador *fix*.

b. Mostrar, si existen, dos términos $M_1, M_2 \in \lambda^{b,n,fix}$ tal que:

i) $\Gamma_1 \triangleright M_1 : \sigma \rightarrow \tau, \Gamma_2 \triangleright M_2 : \sigma$ y $M_1 M_2$ no es tipable. En caso contrario, justificar.

ii) $\exists V_1, V_2$ tal que $M_1 \twoheadrightarrow V_1, M_2 \twoheadrightarrow V_2$, pero no existe V_3 tal que $M_1 M_2 \twoheadrightarrow V_3$. En caso contrario, justificar.

c. Considerar al algoritmo de unificación. Mostrar con un ejemplo que si se eliminan la condición $s \notin FV(\sigma)$ de la regla **Eliminación de variable** y la regla **Occur check** el algoritmo que se obtiene no es correcto.



1
2

① a)

I.

nodes :: LI Int

nodes = \x → x ✓

1	2	3
B	B	B

(A)

II.

nodes :: LI Int

nodes = \x → ~~2 * x~~ 2 * x ✓

III.

repeatI :: [a] → LI a

repeatI xs = \x → ~~xs~~ xs !! (x `mod` (length xs))

b)

I.

(1) consI :: a → LI a → LI a

consI x f = \i → if i == 0 then x
else f(i-1) ✓

(2) headI :: LI a → a

head f = f 0 ✓

(3) tailI :: LI a → LI a

tailI f = \i → f(i+1) ✓

11 ~~Multiple choice (11/11/11)~~

(2) a)

$$(I) \quad T\text{-comp} \quad \frac{\Gamma \vdash S : [M] \quad \Gamma \cup \{x : \tau\} \vdash M : \sigma \quad \Gamma \cup \{x : \tau\} \vdash P : \text{Bool}}{\Gamma \vdash [M | x \leftarrow S, P] : [\sigma]} \quad \checkmark$$

(II)

$$T\text{-comp} \quad \frac{\emptyset \vdash [\lambda x : \text{Nat}. m | m \leftarrow 0 :: 1 :: \square_{\text{Nat}}, \text{even}(m)] : [\text{Nat} \rightarrow \text{Nat}]}{\quad} \quad \checkmark$$

$$T\text{-app} \quad \frac{\emptyset \vdash 0 :: 1 :: \square_{\text{Nat}} : [\text{Nat}]}{\quad} \quad \{m : \text{Nat}\} \vdash \lambda x : \text{Nat}. m : \text{Nat} \rightarrow \text{Nat} \quad (*)$$

$$T\text{-zero} \quad \frac{\emptyset \vdash 0 : \text{Nat} \quad \emptyset \vdash 1 :: \square_{\text{Nat}} : [\text{Nat}]}{\quad} \quad \{m : \text{Nat}\} \vdash \text{even}(m) : \text{Bool}$$

$$T\text{-sw} \quad \frac{\emptyset \vdash 1 : \text{Nat} \quad \emptyset \vdash \square_{\text{Nat}} : [\text{Nat}]}{\quad} \quad T\text{-app} \quad \frac{\{m : \text{Nat}\} \vdash m : \text{Nat}}{\quad} \quad T\text{-zero} \quad \frac{\emptyset \vdash 0 : \text{Nat}}{\quad} \quad T\text{-even} \quad \frac{\{m : \text{Nat}\} \vdash m : \text{Nat}}{m : \text{Nat} \in \{m : \text{Nat}\}} \quad \checkmark$$

(*)

T-abstract

$$T\text{-concrete} \quad \frac{\{m : \text{Nat}, x : \text{Nat}\} \vdash m : \text{Nat}}{m : \text{Nat} \in \{m : \text{Nat}, x : \text{Nat}\}} \quad \checkmark$$

b) (I) $V ::= \dots | \square_{\sigma} | V :: V$

$$E\text{-comp1} \quad \frac{S \rightarrow S'}{[M | x \leftarrow S, P] \rightarrow [M | x \leftarrow S', P]} \quad \checkmark$$

M y P no reducen aún, ya que ~~estas~~ pueden tener a x libre.

E-comp2

$$[M | x \leftarrow V_1 :: V_2, P] \rightarrow \text{if } P \{x \leftarrow V_1\} \text{ then } \dots \text{ else } [M | x \leftarrow V_2, P] \quad \checkmark$$

E-copr₃

$\{x:\sigma\} \cup \Pi \Delta M:\tau$ Para algún Π

$$\frac{}{[M \mid x \leftarrow []_\sigma, P] \rightarrow []_\tau}$$

(I)

$$[\lambda x:\text{Nat}. m \mid m \leftarrow 0 :: 1 :: []_{\text{Nat}}, \text{even}(m)]$$

E-copr₂,
E-iftrue else,
E-even

$$(\lambda x:\text{Nat}. 0) :: [\lambda x:\text{Nat}. m \mid m \leftarrow 1 :: []_{\text{Nat}}, \text{even}(m)]$$

$$(\lambda x:\text{Nat}. 0) :: [\lambda x:\text{Nat}. m \mid m \leftarrow []_{\text{Nat}}, \text{even}(m)]$$

E-copr₃,
E-iftrue else,
E-even

$$(\lambda x:\text{Nat}. 0) :: []_{\text{Nat} \rightarrow \text{Nat}}$$

$$c) \text{ filter}_\sigma (f, xs) \stackrel{\text{def}}{=} [x \mid x \leftarrow xs, f x]$$

$$(II) \text{ map}_{\sigma\tau} (f, xs) = [f x \mid x \leftarrow xs, \text{true}]$$

Están bien pero los macros no son funciones con parámetros. los tenés que definir como funciones lambda.

$$\text{filter}_\sigma \stackrel{\text{def}}{=} \lambda f:\sigma \rightarrow \text{Bool}. \lambda p^\sigma: [\sigma]. \dots$$

$$\text{map}_{\sigma\tau} \stackrel{\text{def}}{=} \lambda f:\sigma \rightarrow \tau. \lambda xs: [\sigma]. \dots$$

(3) a) Supongo extendido el algoritmo de MGV con los nuevos tipos

$$\bullet IW([M, N]) = S\Gamma_1 \cup S\Gamma_2 \supset [SM', SN'] \bullet : \text{Intervalo} \checkmark$$

Si:

$$\begin{aligned} IW(M) &= \Gamma_1 \supset M' : \sigma \checkmark & S &= MGV(\{\sigma = \tau, \sigma = N\tau\}, \\ IW(N) &= \Gamma_2 \supset N' : \tau \checkmark & & \{\sigma_1 = \sigma_2 \mid x : \sigma_1 \in \Gamma_1, \\ & & & x : \sigma_2 \in \Gamma_2\}) \end{aligned}$$

$$\bullet IW(\text{case } M \text{ of } [] \rightsquigarrow N; x :: y \rightsquigarrow O) = *_1$$

Si:

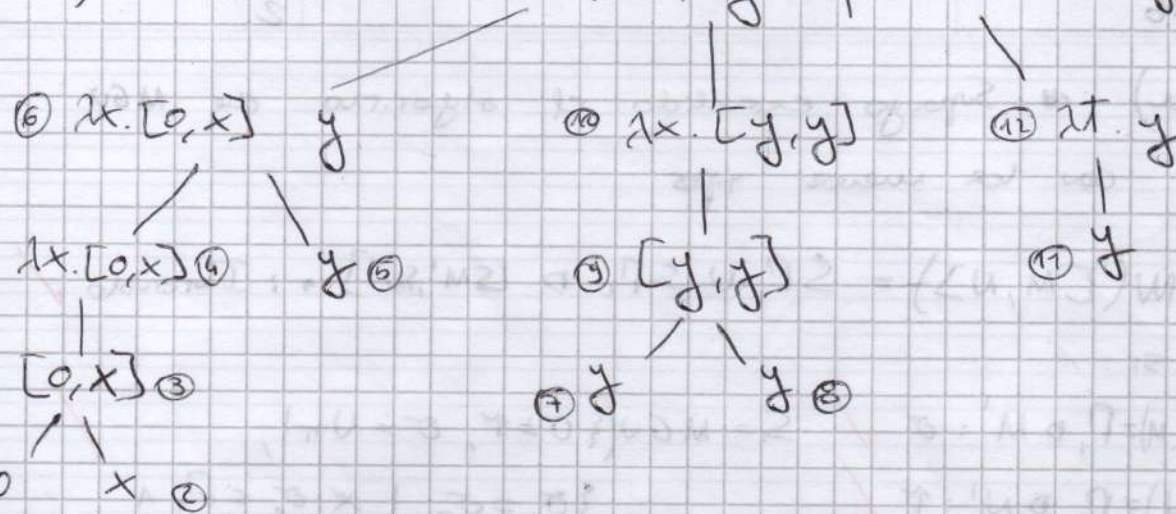
$$\begin{aligned} IW(M) &= \Gamma_1 \supset M' : \sigma \checkmark & \Pi_x &= \begin{cases} \alpha & \text{si } x : \alpha \in \Gamma_3 \checkmark \\ \text{var. fresca} & \text{si no} \end{cases} \\ IW(N) &= \Gamma_2 \supset N' : \tau \checkmark & & \\ IW(O) &= \Gamma_3 \supset O' : \rho \checkmark & \Pi_y &= \begin{cases} \beta & \text{si } y : \beta \in \Gamma_3 \checkmark \\ \text{var. fresca} & \text{si no} \end{cases} \end{aligned}$$

$$\cancel{\Gamma_3'} \quad \Gamma_3' = \Gamma_3 \oplus \{x, y\}$$

$$\begin{aligned} S &= MGV(\{\tau = \rho, \sigma = \text{Intervalo}, \Pi_x = \text{Nat}, \Pi_y = \text{Intervalo} \\ &\quad \cup \{\sigma_1 = \sigma_2 \mid x : \sigma_1 \in \Gamma_i, x : \sigma_2 \in \Gamma_j, \\ &\quad i, j \in \{1, 2, 3\}\}) \checkmark \end{aligned}$$

$$\begin{aligned} *_1 &= S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_3' \supset \text{case } SM' \text{ of } [] \rightsquigarrow SN'; \\ &\quad x :: y \rightsquigarrow SO' : S\tau \checkmark \end{aligned}$$

b) ⑬ case $(\lambda x. [0, x]) y$ of $[] \rightsquigarrow (\lambda x. [y, y]); x :: y \rightsquigarrow \text{Int.y}$



① $|W(0) = \emptyset \triangleright 0 : \text{Nat}$ ✓

② $|W(x) = \{x : s_1\} \triangleright x : s_1$ ✓

③ $|W([0, x]) = \{x : \text{Nat}\} \triangleright [0, x] : \text{Interval}$ ✓
 $s_1 = \text{MGU} \{s_1 = \text{Nat}\} = \{s_1 \in \text{Nat}\}$

④ $|W(\lambda x. [0, x]) = \emptyset \triangleright \lambda x : \text{Nat}. [0, x] : \text{Nat} \rightarrow \text{Interval}$ ✓

⑤ $|W(y) = \{y : s_2\} \triangleright y : s_2$ ✓

⑥ $|W(\lambda x. [0, x] y) = \{y : \text{Nat}\} \triangleright \lambda x : \text{Nat}. [0, x] y : \text{Interval}$ ✓
 $s_2 = \text{MGU} \{s_2 = \text{Nat}\} = \{s_2 \in \text{Nat}\}$ ✓

⑦ $|W(y) = \{y : s_3\} \triangleright y : s_3$ ✓

⑧ $|W(y) = \{y : s_4\} \triangleright y : s_4$ ✓

⑨ $|W([y, y]) = \{y : \text{Nat}\} \triangleright [y, y] : \text{Interval}$ ✓
 $s_3 = \text{MGU} \{s_3 = \text{Nat}, s_4 = s_3\} = \{s_3 \in \text{Nat}, s_4 \in \text{Nat}\}$

⑩ $|W(\lambda x. [y, y]) = \{y : \text{Nat}\} \triangleright \lambda x : s_5. [y, y] : s_5 \rightarrow \text{Interval}$ ✓

⑪ $|W(y) = \{y : s_6\} \triangleright y : s_6$ ✓

⑫ $|W(\text{Int.y}) = \{y : s_7\} \triangleright \text{Int.y} : s_7 \rightarrow s_6$ ✓

4
2

⑬ $W(\text{case } (\lambda x. [0, x]) \text{ of } [] \leadsto (\lambda x. [y, y]));$
 $x :: y \leadsto \lambda t. y) = *_2$

Tipos

$$W(\lambda x. [0, x]) = \{y : \text{Nat}\} \triangleright \lambda x : \text{Nat}. [0, x] y : \text{Intervalo}$$

$$W(\lambda x. [y, y]) = \{y : \text{Nat}\} \triangleright \lambda x : S_5. [y, y] : S_5 \rightarrow \text{Intervalo}$$

~~$W(\lambda x. [y, y])$~~

$$W(\lambda t. y) = \{y : S_6\} \triangleright \lambda t : S_7. y : S_7 \rightarrow S_6$$

$$S_4 = \text{MGU}(\{ \text{Intervalo} = \text{Intervalo}, S_5 \rightarrow \text{Intervalo} = S_7 \rightarrow S_6, \\ S_6 = \text{Intervalo} \} \cup \{ \text{Nat} = \text{Nat} \}) = \\ = \{ S_5 \leftarrow S_7, S_6 \leftarrow \text{Intervalo} \}$$

$$\mathcal{P} = S_4 \{y : \text{Nat}\} \cup S_5 \{y : \text{Nat}\} \cup S_6 \emptyset = \{y : \text{Nat}\}$$

$$*_2 = \mathcal{P} \triangleright (\text{case } \lambda x : \text{Nat}. [0, x] y \text{ of } [] \leadsto \lambda x : S_7. [y, y]; \\ x :: y \leadsto \lambda t : S_7. y) : S_7 \rightarrow \text{Intervalo}$$

Veo que hay un problema con el contexto λy la y de $\lambda t : S_7. y$. Por α -equivalencia daría lo mismo que en lugar de $x :: y$ el case tuviera $x :: z$, por lo tanto λy , al hacer $\mathcal{P} \cup \{y : \text{Intervalo}\}$ no debería haber problema, ya que podría ser $\mathcal{P} \cup \{z : \text{Intervalo}\}$. Sinceramente, no recuerdo cómo se completaba ese caso en el algoritmo, ~~pero~~ pero se podría realizar ese chequeo al cambiar y reemplazar x e y en $x :: y$ por variables que no aparezcan libres en M y N . (*) Ver atrás

⊛ Acordate que en la regla que definiste unificás las variables que aparecen en más de un contexto pero para el contexto de \emptyset previamente le sacás las variables x e y (cuando hacés $\Gamma'_3 = \Gamma_3 \ominus \{x, y\}$). Así que este conflicto no existe.